モデル間の予測誤差を利用した効率的な強化学習手法

橋本 大世[1,a)]　鶴岡 慶雅[2]

概要：強化学習は囲碁などのボードゲームや Atari 2600 などのビデオゲームで多くの成功を収めているが, 教師あり機械学習などと比べると未だに実社会での応用例は限られている. この理由の一つとして, サンプル効率の低さが挙げられる. また, 現実的なタスクでは報酬がスパースになりやすいが, 特にそのような環境では効率的な学習は難しい. 本論文では, 報酬がスパースな環境においても効率的に学習することのできる強化学習手法を提案する. 具体的には, モデルベース強化学習と内部報酬を組み合わせることで, 環境の探索および方策の学習を効率化する. また, 少量の画像から方策を学習するために, 画像をランダムにエンコードするという手法を考案する. 本稿では OpenAI Gym の MountainCar と Freeway において実験を行い, 画像を入力とする場合においても, 簡単なものであれば効率的な学習が可能であることを示した.

An Efficient Reinforcement Learning Method Using Prediction Errors Between Models

Taisei Hashimoto[1,a)]　Yoshimasa Tsuruoka[2]

Abstract: Reinforcement learning has been successful in board games such as Go and video games such as Atari 2600, but its application in the real world is still limited compared to supervised machine learning. One of the reasons is its low sample efficiency. Moreover, the rewards tend to be sparse in realistic tasks, and efficient learning is difficult especially in such an environment. In this study, we propose a reinforcement learning method that can learn efficiently even with sparse rewards. Specifically, we make the environment exploration and policy learning more efficient by combining model-based reinforcement learning and intrinsic rewards. Also, we have devised a method to learn a policy from a small number of image observations by randomly encoding them. In this paper, we conducted experiments with MountainCar and Freeway of OpenAI Gym and verified that effective learning is possible also from raw images as long as they are simple.

1. はじめに

近年, 強化学習は深層学習などの技術により様々な成果をあげている. 例えば, Atari 2600 などのゲームにおいて, 深層強化学習を用いたゲーム AI が人間を上回るスコアを出すことができるようになってきている [1], [2].

このような成果があるにもかかわらず, 教師あり機械学習に比べて強化学習が実社会で応用されている例は少ない. この理由として, 強化学習におけるいくつかの課題が影響していると考えられる.

まず, 強化学習は一般的にサンプル効率が高くない. 教師あり機械学習では入力に対して答えが与えられてそれらの関係性を学習するが, 強化学習では直接的な答えが与えられるわけではない. そのため, 試行錯誤しながら各状態に対してどう行動するべきかを学習する必要があり, 多くのサンプルを必要とする. 例えば画像を入力とするゲーム AI では, 学習に数 100M フレーム以上を必要とすることも珍しくない.

この問題の対処法として, 環境のモデルを学習して利用するという方法がある. この手法はモデルベース強化学習と呼ばれ, モデルを用いないモデルフリー強化学習と対比される. モデルベース強化学習では環境から収集したデー

[1] 東京大学工学部電子情報工学科
Department of Information and Communication Engineering, The University of Tokyo
[2] 東京大学大学院情報理工学系研究科電子情報学専攻
Department of Information and Communication Engineering, Graduate School of Information Science and Technology, The University of Tokyo
[a)] hashi@logos.t.u-tokyo.ac.jp

タを使ってモデルを学習し, 得られたモデルを方策の学習に利用する. 正確なモデルを得られれば実際の環境に頼らず方策を学習できるため, サンプル効率を高めることができる.

Hafner らは Planning Network (PlaNet) という手法を提案した [3]. PlaNet は画像入力からその後のフレームを予測し, 累積報酬を最大化するよう行動をプランニングする. この手法により, いくつかの制御タスクにおいて A3C [4] や D4PG [5] といったモデルフリーの手法よりもかなり高いサンプル効率で近いパフォーマンスを達成した.

また, 一般的な強化学習手法では報酬がスパースな環境下での学習が難しいという課題もある. 強化学習用の環境の中には報酬が頻繁に与えられるものもあるが, 実社会での利用を考えると, 「このタスクを達成したら +1 の報酬」といった報酬設計が望ましく, 自然と報酬はスパースになる. タスクに応じて上手に報酬を設計することも可能ではあるが, コストがかかるだけでなく, 設計の仕方が良くないとエージェントが想定外の方策を学習してしまう可能性もある.

この問題には, 環境から与えられる報酬とは別に内部報酬と呼ばれる報酬をエージェントに与え, 効率的な探索を促すという手法がある. 内部報酬の生成方法は様々だが, 今まで多く経験していない状態を優先的に探索するよう動機づけするという点は共通している.

Burda らは Random Network Distillation (RND) という手法を提案した [6]. RND はランダムに初期化されたニューラルネットワークに現在の状態を入力し, その出力をもう一つのニューラルネットワークで予測して, 予測誤差を内部報酬とする. この手法により, Atari 2600 の中で最難関と言われる Montezuma's Revenge において当時の最高記録を更新し, 人間の平均を超えるスコアを上回った.

本研究の目的は, モデルベース強化学習と内部報酬を組み合わせることで, 報酬がスパースな環境においても効率よく学習可能な強化学習手法を考案することである. 同様の発想に基づく手法はすでに存在する [7] が, 2.3 項で述べるようにいくつかの問題点がある.

そこで, 我々はより汎用的な手法を提案し, いくつかのタスクをに適用することでその有効性の実証を試みた.

特に観測データとして画像を利用する環境では, 画像をランダムにエンコードするというアプローチにより, サンプル効率と計算コストの改善を試みた.

2. 関連研究

2.1 モデルベース強化学習

モデルベース強化学習では一般に, 実環境から得られたデータにより環境モデルを学習し, そのモデルを利用してエージェントがより高い報酬を得られるように方策を改善していく. 環境モデルの具体的な利用方法には大きく分けて 2 つある. 一つはモデルにより各行動に対する累積報酬を予測し, 報酬が最大化される行動を選択するものであり, もう一つはモデルフリー強化学習手法を使ってモデル内で方策を学習するものである.

これまでに, モデルベース強化学習を用いて学習効率を向上させた研究は多数存在する.

Hafner らは Planning Network (PlaNet) という手法を提案した [3]. PlaNet では Variational Autoencoder (VAE) [8] のように潜在空間を用いて画像から確率的な遷移モデルを学習し, 報酬の予測に利用する.

この論文では, 確率的な遷移だけでなく確定的な遷移を含むモデルを構築している. 具体的には, 潜在空間内で状態が確率的に遷移する一方, 状態を入力として隠れ状態を再帰的かつ確定的に計算する. こうすることで, 純粋に確率的な遷移モデルに比べて過去の情報が失われにくいため, 予測性能が向上したと報告されている.

同様の主張は, 画像からモデルを作成した他のいくつかの論文でも見られる [9].

Ha らは純粋な VAE を学習し, それによって得られた潜在空間における遷移モデルを構築し, モデル内で方策を学習した [10]. 画像入力をエンコードするビューモデル, 潜在空間における遷移モデル, 方策を決めるコントローラモデルがそれぞれ分離されており, 学習しやすいという特徴がある.

遷移モデルでは PlaNet と同様に隠れ状態を再帰的に計算し, それを利用して次状態を予測する. この手法では, 各状態は対応する画像を単純に VAE でエンコードしたものなので, 状態 1 つだけでは速度などの情報が失われてしまう. そのため隠れ状態を利用することは必須であると考えられる.

また PlaNet などの確率的な遷移モデルを使う他の手法の多くでは, 確率分布としてガウス分布を用いているが, Ha らは混合ガウス分布を用いることで複雑なモデルの学習にも対応した.

Kaiser らは Simulated Policy Learning (SimPLe) という手法を提案した [11]. SimPLe は上述の 2 手法とは異なり, 現在の画像から直接次の画像を予測する. 画像 1 枚では速度などの情報が得られないため, 画像を数枚スタックして入力として用いている.

2.2 内部報酬

報酬がスパースな環境では, 探索の効率を向上させるために内部報酬を利用する手法がある.

Pathak らは Intrinsic Curiosity Module (ICM) という機構を提案した [12]. ICM は各状態に特徴量ベクトルを対応させ, そのベクトルの予測モデルを学習し, 予測誤差を内部報酬とする. ある状態を何度も経験していれば, 次状態の特徴量ベクトルはある程度正確に予測できるため, 内部報

酬は小さくなる．一方あまり経験していない状態であれば，内部報酬は大きくなる．このようにして，新しい状態を優先的に探索するよう動機づけを行う．

Burda らは ICM の問題点を指摘し，Random Network Distillation (RND) という手法を提案した [6]．ICM では間接的に 1 タイムステップ先の状態を予測する必要があるが，環境がランダム性を持つ場合には完全な予測は不可能である．そのため，環境のランダムな部分に対していつまでも高い内部報酬を生成し，有効な探索ができなくなってしまう問題がある．この問題は一般に noisy-TV problem と呼ばれる [13]．

そこで Burda らは，ランダムに初期化されたニューラルネットワークに現在の状態を入力し，その出力をもう一つのニューラルネットワークで予測して，その予測誤差を内部報酬とした．この手法により，Atari 2600 の中で最難関と言われる Montezuma's Revenge において，当時の最高記録を更新した．

2.3 モデルベース強化学習と内部報酬の組み合わせ

Chua らは SU-PETS という手法を提案した [7]．この手法は unsupervised フェーズと goal-directed フェーズの 2 つのフェーズからなる．unsupervised フェーズでは外部報酬が与えられず，エージェントは純粋な探索により環境モデルを学習する．その際，複数の確率的な環境モデルを学習し，それらのモデルの予測誤差を内部報酬として利用する．環境がランダム性を持つ場合でも各環境モデルの予測は近い確率分布になることが期待でき，これにより noisy-TV problem を緩和することができる．goal-directed フェーズでは，候補となる複数の行動系列に関して学習済みの環境モデルにより状態の予測を行い，累積報酬の期待値が最も高くなる行動を選択する．

SU-PETS は OpenAI Gym [14] の HalfCheetah という環境ではわずか数十エピソードで高いスコアに到達しているが，いくつかの問題点がある．

まず，goal-directed フェーズにおいて真の報酬関数を利用している．通常真の報酬関数は利用できず，環境から報酬モデルを学習するか，モデルフリー強化学習のように価値関数を学習する必要がある．そのためこの手法は，環境の報酬関数は既知だが遷移モデルは未知という状況でしか有効活用できず，汎用性が高いとは言えない．

また，モデルをプランニングに使っており方策は学習しないため，十分先まで予測しなければ，近視眼的な行動をとってしまう可能性がある．これは報酬がスパースな環境では特に問題である．強化学習において特に効率的探索が重要になるのは報酬がスパースな環境であるにもかかわらず，SU-PETS はそのような環境にはあまり適さないと考えられる．実際，HalfCheetah の環境ではエージェントの速度などに応じて毎ステップ報酬が与えられ，報酬がスパースな環境とは言えない．

そして，SU-PETS は HalfCheetah 以外での実験がなされておらず，他の環境で機能するかは検証が必要だろう．特に最近は画像などの高次元データからモデルを学習する手法も増え [3], [10]，そういった学習が可能かどうかも調査する価値がある．

3. 提案手法

3.1 概要

我々は複数のモデルの予測誤差を内部報酬として利用し，モデル内で方策を学習する手法を考案した．提案手法の擬似コードを Algorithm 1 に示す．エージェントは，外部報酬に加えて内部報酬を多く得られる方策，すなわちモデルの学習が進んでいない状態に到達することのできる方策をモデル内で学習し，その方策に従って実際の環境でデータを収集する．

3.2 画像のエンコード

2.1 項で述べたように，画像が入力として与えられる環境では画像から直接次の画像を予測するか，画像を潜在空間にエンコードして潜在空間内の遷移を予測するかの 2 種類のモデルが考えられる．どちらもモデルの訓練時には画像を予測して実際の画像と比べるため，計算コストが高い．特に直接画像を予測する場合は，モデル内で方策を学習する際も画像を毎ステップ出力する必要がある．

また潜在空間にエンコードする場合は，状態の遷移だけでなくエンコードとデコードについても学習することになる．適切な状態空間を生成するようになるにはある程度のサンプルが必要であり，さらに学習中に潜在空間の形状が変わることにより，モデルの学習速度が遅くなる可能性がある．

そもそもタスクを解くという観点では，画像を正確に予測することが必要であるかどうかは自明ではない．Dubey らは人間の事前知識がビデオゲームのプレーにどのように影響しているのかを調査した [15]．その結果，事前知識を用いることができないようにゲームの画像を変化させると，人間は学習に多くの時間が必要となったのに対し，強化学習エージェントはほとんど同様に学習できることがわかった．この研究から，強化学習エージェントにとってゲーム画面は我々が思うほど特別な，理解しやすい表現形式ではないと言える．

そこで我々は，画像をランダムにエンコードし，その結果生成されるベクトルの空間における遷移を学習した．具体的には，ランダムに初期化された畳み込みニューラルネットワークに画像を入力し，その出力を 1 次元の実ベクトルに変形する．これを中間表現と呼ぶことにする．

中間表現は画像をランダムにエンコードしたものなので，かなり冗長な表現になる．このままではベクトルの次元が

高く, 予測精度の悪化や計算コストの増大につながるため, これを主成分分析 (PCA) により次元削減する. その結果得られたベクトルを, ここではエンコード表現と呼ぶことにする.

通常の PCA では一度にすべてのデータが与えられるが, 今回の場合はデータが環境から次々に与えられて主成分を更新していく必要がある. そこで, Incremental PCA [16] と呼ばれる手法を利用した. Incremental PCA はサイズが大きくデータ全体に対する PCA ができない場合に用いられることが多く, データをミニバッチに分けてバッチごとに計算を行い, 主成分を更新する.

以降, 入力が画像で与えられる環境での観測データはエンコード表現を指す. なお入力がエージェントの位置などの特徴量で与えられる環境では, このような変換は必要ない.

3.3 環境モデル

実際の環境をシミュレートするために, 時刻 t の観測データ $o_t \in \mathbb{R}^d$ と行動 $a_t \in \{1, 2, \dots n\}$ から, 次の観測データ o_{t+1} を予測するモデル, 報酬 $r_t \in \mathbb{R}$ を予測するモデル, 終了状態 (終了したかどうか) $d_t \in \{\text{True}, \text{False}\}$ を予測するモデルを用意する. d は観測データの次元, n は行動の選択肢の個数である. ここではそれぞれのモデルを観測モデル, 報酬モデル, 終了モデルと呼ぶことにする. これらのモデルはいずれもニューラルネットワークで実装され, 3 つ合わせて環境モデルとなる.

観測モデル

PlaNet や Ha らの手法に倣い, 観測モデルは次の観測データの確率分布をガウス分布として予測することにした.

観測モデルの構成は, 観測データの種類によって異なる.

観測データがエージェントの位置などの特徴量の場合は, 単純に o_t, a_t から o_t の確率分布を予測する.

一方, 観測データが画像の場合は 1 枚の画像からは完全に状態を知ることはできないので, 2.1 項で述べた PlaNet や Ha らの手法の様に, 隠れ状態 h_t を再帰的に計算して予測に利用する.

$$h_{t+1} = f_h(h_t, o_t, a_t)$$

f_h は RNN で実装され, 確定的な遷移を行う.

隠れ状態を使用しない場合と使用する場合それぞれにおけるデータの流れを図 1, 2 に示す.

以降では便宜上入力の h_t を省略して表記するが, 観測データが画像の場合は常に o_t とともに h_t が与えられる.

観測モデルは次の観測データの平均, 標準偏差をベクトルとして予測する.

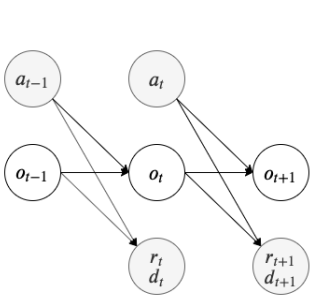

図 1 データフロー図 (隠れ状態なし)

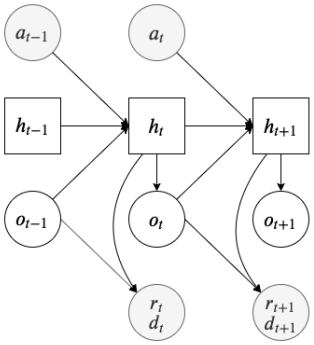

図 2 データフロー図 (隠れ状態あり)

$$\mu_{t+1} = f_\mu(o_t, a_t)$$
$$\sigma_{t+1} = f_\sigma(o_t, a_t)$$

観測モデルの損失関数を

$$\mathcal{L}_o = -\log p(o_{t+1}; \mu_{t+1}, \sigma_{t+1})$$

と定義する.

報酬モデル

報酬モデルは o_t, a_t から, a_t の結果として得られる r_{t+1} を予測する.

報酬モデルの損失関数を,

$$\mathcal{L}_r = (r_{t+1} - f_r(o_t, a_t))^2$$

と定義する.

終了モデル

終了モデルは o_t, a_t から, a_t の結果としてエピソードが終了する確率, すなわち $d_{t+1} = \text{True}$ となる確率を出力する.

終了モデルの損失関数を,

$$\mathcal{L}_d = \text{cross\_entropy}(f_d(o_t, a_t), d_t)$$

と定義する. ただし, $\text{cross\_entropy}(x, y)$ は予測 x, 正解ラベル y に関する交差エントロピーを表す.

提案手法では, 内部報酬を生成するために観測モデルを複数用意した. そのため, 観測モデルの損失関数は各モデルについて計算され, それらを平均したものを観測モデル全体の損失 L_o とする.

環境モデル全体の損失関数は

$$\mathcal{L}_M = c_o\mathcal{L}_o + c_r\mathcal{L}_r + c_d\mathcal{L}_d \tag{1}$$

と定義する. ここで, c_o, c_r, c_d はそれぞれの損失関数の係数である.

環境モデルでシミュレーションを行う際, 複数の観測モデルが現在の状態から次状態を再帰的に予測していくが, このときの予測の標準偏差を内部報酬とした. より具体

アルゴリズム 1 提案手法

Inputs:

K	Number of models
C	Interval of model update
α	Learning rate for model parameters
β	Learning rate for policy parameters

Initialize empty data set $\mathcal{D}$
Initialize model parameters ϕ randomly
Initialize policy parameters θ randomly
while not converged **do**
  // Model fitting
  Collect data from real environment.
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(o_t, a_t, r_t, d_t)_{t=1}^{T}\}$
  Compute loss $\mathcal{L}_M(\phi)$ from Equation 1.
  Update model parameters $\phi \leftarrow \phi - \alpha \nabla \mathcal{L}_M(\phi)$.
  // Policy improvement
  for update step $t = 1..C$ **do**
    Generate data set $\mathcal{D}'$ from model simulation.
    Compute PPO objective $\mathcal{L}_P(\theta)$.
    Update policy parameters $\theta \leftarrow \theta + \beta \nabla \mathcal{L}_P(\theta)$.
  end for
end while

的には, 各観測モデルの出力 μ_t, σ_t を結合したベクトルに $\mu_t|\sigma_t$ について, 次元ごとの標準偏差を計算し, その総和を内部報酬としている. なお, 内部報酬はそれまでに生成された内部報酬の標準偏差で割ることにより正規化している.

3.4 方策

エージェントの方策は, 環境モデル内で Proximal Policy Optimization (PPO) [17] により学習する. 報酬としては, 内部報酬と外部報酬の単純な和を用いる. 環境モデルの学習が不十分なうちは内部報酬が大きく, 環境の探索が優先されると考えられる. モデルの学習が進むと内部報酬が小さくなっていき, 外部報酬を高めるような方策を習得していくことが期待される.

4. 実験

4.1 実験設定

4.1.1 環境

提案手法の有用性を検証するため, OpenAI Gym の MountainCar 及び Freeway という 2 つのタスクにおいて, 提案手法により学習を行った.

いずれの環境においても RND や SimPLe に倣ってスキップ数 4 のフレームスキップを適用した. すなわち, 1 つの行動を 4 回繰り返し, その後 1 つの観測データを得るという処理を行っている. 以降ステップと言う場合, スキップなしの環境におけるステップ数を指す. つまり, エージェントにとっての 1 行動が 4 ステップに対応する.

MountainCar

図 3 に MountainCar のプレー画面を示す. このタスクでは, 車を山の頂上にある, 位置 0.5 のゴールに到達させることが目的である. エージェントは左右どちらかに加速するか, 何もしないかの 3 通りの行動を取ることができる. 報酬は常に -1 であり, ゴールするか 200 ステップが経過するとエピソードは終了する. また, 初期状態は位置が -0.6 から -0.4 までのランダムな値で, 速度は 0 である.

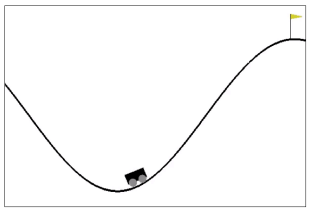

図 3 MountainCar のプレー画面

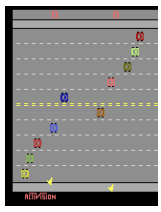

図 4 Freeway のプレー画面

このタスクは簡単なタスクとして有名であり, 単純な Q 学習 [18] などでも解くことは可能である. しかし, 常に報酬が -1 であることは報酬がスパースであることと近い意味を持っており, 少ないサンプル数でゴールにたどり着くためには効率的な探索手法が必要となる.

この環境では本来, 観測データとして車の位置と速度が実数値で与えられる. 本研究では, 位置などの特徴量が与えられる場合と画像が与えられる場合の学習の様子を比較するため, 特徴量を入力とした学習に加えて画像を入力とした学習も行った.

Freeway

図 4 に Freeway のプレー画面を示す. このタスクでは, 手前の歩道にいるニワトリが車に当たらないようにしながら奥の歩道に向かうことが目的である. 車に当たった場合には少し手前に戻されるが, ゲームオーバーになったり負の報酬が与えられたりはしない. 観測データとしてはゲーム画面が与えられる. エージェントは前後どちらかに 1 歩進むか, 何もしないかの 3 通りの行動を取ることができる. 報酬はゴールしたときに 1, それ以外は 0 であり, 8192 ステップ経過するとエピソードは終了する.

ゴールまで到達しなければ一切の報酬は得られないため, この環境は報酬がスパースと言える.

4.1.2 モデル設計

観測モデル, 報酬モデル, 終了モデルは 1 層のニューラルネットワークで実装した. 隠れ状態がない場合とある場合の構成を, それぞれ図 5, 6 に示す. 図中の FC は全結合層, Embed は埋め込み層, RNN は再帰型ニューラルネットワークを意味する. RNN としては, GRU [19] を用いた. なお, 隠れ状態が次の観測の予測のみに利用されるように促すため, 報酬モデルや終了モデルの損失から生じた勾配は, 隠れ状態には流していない.

エージェントの Policy ネットワーク, Value ネットワー

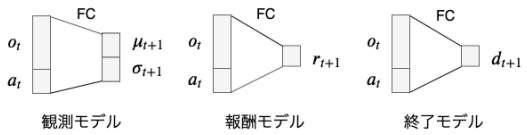

図 5　モデルの構成
(隠れ状態なし)

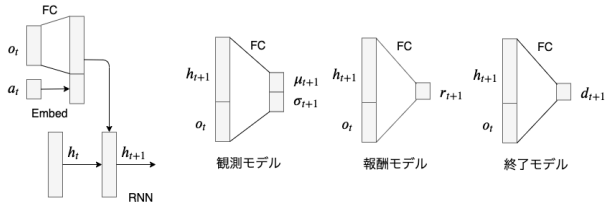

図 6　モデルの構成
(隠れ状態あり)

クはいずれも 2 層のニューラルネットワークで実装し, 隠れ層の次元は 200 とした. また, エージェントの Critic ネットワークでは RND の実装にならい, 外部報酬と内部報酬の出力を別々に用意した.

すべてのニューラルネットワークについて, 活性化関数としては ReLU を用いた.

4.1.3　学習方法

基本的に 3 節で説明したとおりに学習を行った. ただし, 観測データが画像でない場合は次元ごとのスケールが異なり, 学習に悪影響を及ぼす. そこで, 各時点までに得られた観測値の平均, 標準偏差を計算し, 観測データを標準化した. 提案手法では観測値の予測の標準偏差を内部報酬として使用しているため, この処理は特に重要である.

この手法では, モデルの予測が正確でない状態から優先的に学習することが重要である. そうしなければモデルの学習が進んでいない状態に対して高い内部報酬を長時間生成し続けてしまい, 似たデータを重複して集めることになるからである. これを防ぐため, メモリからデータをランダムにサンプリングするのではなく, Prioritized Experience Replay [20] のようにモデルの予測誤差に応じて優先度を計算し, 学習が進んでいないデータを優先的にサンプリングするようにした.

また, 環境モデル内でのシミュレーションでは初期状態を設定する必要があるが, 初期状態はそれまでに経験した状態からランダムに選ぶことにした.

4.1.4　その他設定

損失関数の係数 c_o, c_r, c_d は, MountainCar ではすべて 1 とした. 一方 Freeway では報酬の損失関数が小さい値となったので, 学習を早めるために c_r は 10 とし, c_o, c_r は 1 とした.

モデル内のシミュレーションのステップ数は, すべての実験において 100 とした.

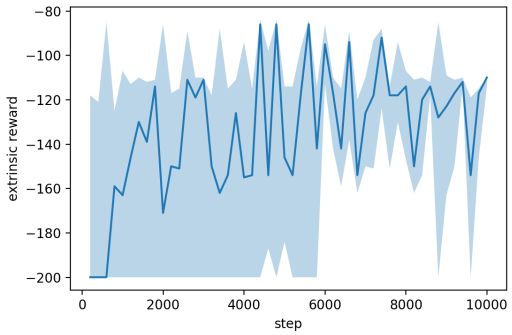

図 7　位置と速度を入力とする MountainCar において, 提案手法により得られたエピソードごとの報酬

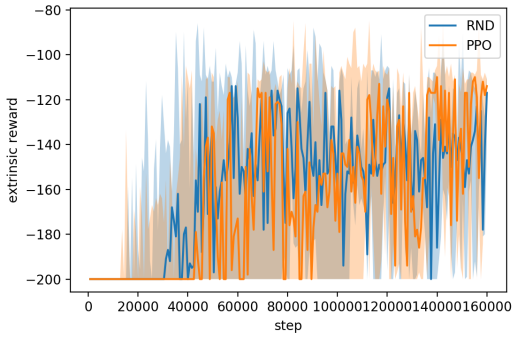

図 8　位置と速度を入力とする MountainCar において, RND および PPO により得られたエピソードごとの報酬

4.1.5　比較対象

比較対象としては, モデル及び内部報酬を使わない PPO エージェントと, RND により生成される内部報酬を利用する PPO エージェントを用意した.

2.2 項で述べたように, RND は内部報酬を利用して効率的な探索を行う. これにより, 純粋な PPO よりも早く学習できると予想される.

4.2　実験結果

4.2.1　MountainCar (入力: 位置と速度)

提案手法, RND, PPO を用いて学習した結果をそれぞれ図 7, 8 に示す. 各手法について 5 回学習しており, 図中の実線は中央値を, 色付きの領域は最小値から最大値までを表す.

提案手法は RND よりも高いサンプル効率で学習できていることがわかる. またこの環境では, RND による学習効率の顕著な向上は見られなかった.

4.2.2　MountainCar (入力: 画像)

提案手法において, シミュレーションのステップ数は 100 とした. 結果を図 9, 10 に示す.

提案手法は RND よりも高いサンプル効率で学習できていることがわかる. 特に RND では一度もゴールに到達しないことがあったのに対し, 提案手法では 5 回全てにおいてゴールに到達しており, 効率的な探索ができている. こ

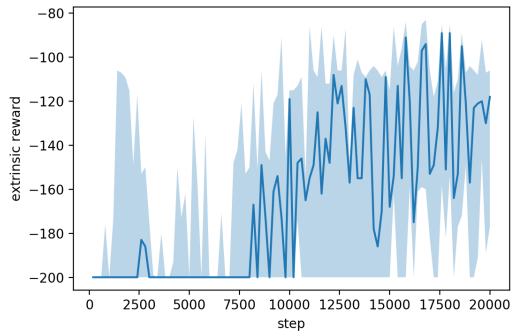

図 9　画像を入力とする MountainCar において, 提案手法により得られたエピソードごとの報酬

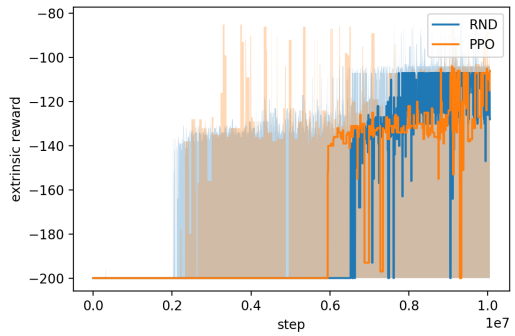

図 10　画像を入力とする MountainCar において, RND および PPO により得られたエピソードごとの報酬

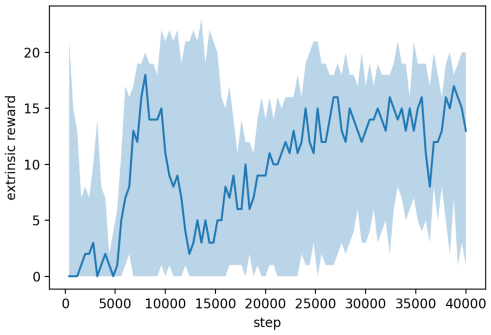

図 11　Freeway において, 提案手法により得られたエピソードごとの報酬

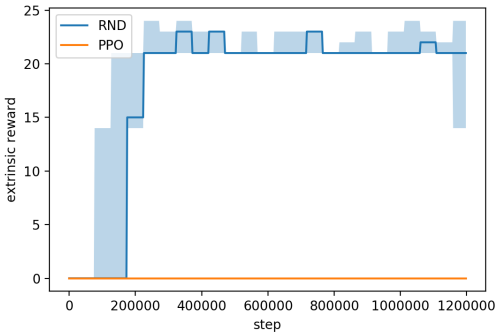

図 12　Freeway において, RND および PPO により得られたエピソードごとの報酬

の環境でも, RND による学習効率の顕著な向上は見られなかった.

4.2.3　Freeway

提案手法において, シミュレーションのステップ数は 100 とした. 結果を図 11, 12 に示す.

提案手法は効率よく探索し, 早い段階でゴールに到達してはいるものの, 最終的なスコアは RND よりも低く, 特に試行ごとの安定性に劣る結果となった. これはモデルの不正確さにより, 方策の学習が進まなかったことによるものと考えられる. この環境では特に報酬の予測が重要であるが, 画像が複雑なためにその部分がうまく学習できなかった可能性がある.

提案手法では学習の過程でモデルの損失関数が急に増大することがあり, そのような場合は学習がうまく進まないことが多かった. これはおそらくモデルの過学習によるものだと思われる. 対処法としてはデータ収集に複数のエージェントを用いて, 集められるデータの偏りを抑えることなどが考えられるが, これは今後の課題としたい.

提案手法による学習中, 特に学習の初期段階において, エージェントが車に近づいて当たりそうになったら避けるという行動が多く見られた. これは, 車に当たるかどうかの判定がその時点では高い精度でできておらず, この行動に高い内部報酬が生成されているためだと考えられる.

PPO はゴールまでたどり着けず, 全く学習できていない. 一方 RND は内部報酬によりすぐにゴールにたどり着き, その後概ね安定して報酬を得られている.

4.3　考察

MountainCar においては, 提案手法は RND よりも高いサンプル効率を達成することができた. この主な要因は, 提案手法はモデルベース強化学習を用いており, モデル内で何度も方策を学習することができるためだと考えられる.

一方, Freeway においては, 提案手法は探索は効率的にできたものの, モデルの学習が難しいためその後の方策の学習がうまく行かないケースが多かった. この点は, 画像をエンコードする方法を修正することにより改善できる可能性もある.

RND による学習効率の向上は, MountainCar においては小さく, Freeway において大きかった. これはおそらく MountainCar の報酬が常に -1 であり, 学習された状態における価値関数は -100 程度の値となるため, 単純な PPO であっても学習の進んでいない方向へ進もうとするためであると考えられる. 一方 Freeway ではゴールまで到達できなかった場合は環境から一切の報酬を受け取れないため, 内部報酬が特に重要な環境だと考えられる.

また, Freeway の探索は常に前進するだけでも可能だが, MountainCar の場合は一度左に動いて勢いをつけてから右に動くという, 少し複雑な動作をする必要があることも影

響したと考えられる．RND は今まで訪れた状態の中で学習の進んでいないものに対しては高い内部報酬を生成するが, 今まで一度も訪れていない状態に対しては, モデルがないため内部報酬を生成できない．そのため, MountainCar での探索方法を少量のサンプルで見出すのは難しかったのだと思われる．

5. おわりに

本稿では, OpenAI Gym のゲームを題材とし, 提案手法の有効性を検証した．その結果, 特に画像を入力とする MountainCar に関しては, 画像をランダムにエンコーディングすることにより高いサンプル効率を達成した．

今後の課題としては, Freeway の実験で見たように複雑な画像から学習する難しさを克服することが挙げられる．また, データ収集に複数のエージェントを用いるなどの改良により提案手法を安定化させるなどの改善を考えている．

参考文献

[1] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. and Riedmiller, M.: Playing Atari With Deep Reinforcement Learning, *NIPS Deep Learning Workshop* (2013).

[2] Hessel, M., Modayil, J., van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M. G. and Silver, D.: Rainbow: Combining Improvements in Deep Reinforcement Learning, *Thirty-Second AAAI Conference on Artificial Intelligence* (2018).

[3] Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H. and Davidson, J.: Learning Latent Dynamics for Planning from Pixels, *arXiv preprint arXiv:1811.04551* (2018).

[4] Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D. and Kavukcuoglu, K.: Asynchronous Methods for Deep Reinforcement Learning, *International Conference on Machine Learning* (2016).

[5] Barth-Maron, G., Hoffman, M. W., Budden, D., Dabney, W., Horgan, D., TB, D., Muldal, A., Heess, N. and Lillicrap, T. P.: Distributed Distributional Deterministic Policy Gradients, *International Conference on Learning Representations* (2019).

[6] Burda, Y., Edwards, H., Storkey, A. and Klimov, O.: Exploration by random network distillation, *International Conference on Learning Representations* (2019).

[7] Chua, K., McAllister, R., Calandra, R. and Levine, S.: Unsupervised Exploration with Deep Model-Based Reinforcement Learning, *Deep Reinforcement Learning Workshop NeurIPS 2018* (2018).

[8] Kingma, D. P. and Welling, M.: Auto-Encoding Variational Bayes, *International Conference on Learning Representations* (2014).

[9] Gregor, K., Papamakarios, G., Besse, F., Buesing, L. and Weber, T.: Temporal Difference Variational Auto-Encoder, *International Conference on Learning Representations* (2019).

[10] Ha, D. and Schmidhuber, J.: Recurrent World Models Facilitate Policy Evolution, *Advances in Neural Information Processing Systems 31* (2018).

[11] Kaiser, L., Babaeizadeh, M., Milos, P., Osinski, B., Campbell, R. H., Czechowski, K., Erhan, D., Finn, C., Kozakowski, P., Levine, S. et al.: Model-based reinforcement learning for atari, *arXiv preprint arXiv:1903.00374* (2019).

[12] Pathak, D., Agrawal, P., Efros, A. A. and Darrell, T.: Curiosity-driven Exploration by Self-supervised Prediction, *International Conference on Machine Learning* (2017).

[13] Burda, Y., Edwards, H., Pathak, D., Storkey, A., Darrell, T. and Efros, A. A.: *International Conference on Learning Representations* (2019).

[14] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J. and Zaremba, W.: OpenAI Gym, *arXiv preprint arXiv:1606.01540* (2016).

[15] Dubey, R., Agrawal, P., Pathak, D., Griffiths, T. L. and Efros, A. A.: Investigating Human Priors for Playing Video Games, *International Conference on Machine Learning* (2018).

[16] Ross, D. A., Lim, J., Lin, R.-S. and Yang, M.-H.: Incremental learning for robust visual tracking, *International journal of computer vision* (2008).

[17] Schulman, J., Wolski, F., Dhariwal, P., Radford, A. and Klimov, O.: Proximal policy optimization algorithms, *arXiv preprint arXiv:1707.06347* (2017).

[18] Watkins, C. J. C. H.: Learning from Delayed Rewards, PhD Thesis, King's College (1989).

[19] Chung, J., Gulcehre, C., Cho, K. and Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling, *arXiv preprint arXiv:1412.3555* (2014).

[20] Schaul, T., Quan, J., Antonoglou, I. and Silver, D.: Prioritized Experience Replay, *International Conference on Learning Representations* (2016).